Date ___/___/_____

Heaps.

Whenever there is a priority linked with anything, we have a data structure which is called Heap or Priority Queue.

Heap is a Complete Binary Tree.*

Types of Heap :-

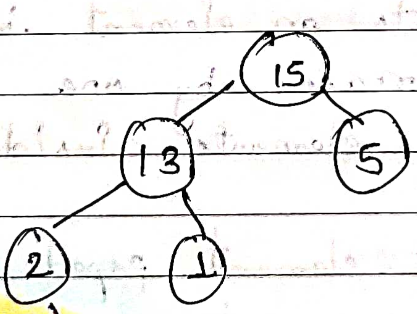
Heap is like a queue but with certain priority
Two types of Heap are there.

(1) Max Heap :- Priority is given to the maximum element. It is sorted in descending order of elements.

ex. [2, 13, 5, 1, 15] → sample element.
Array implementation of Max heap.

Top → [15 | 13 | 5 | 2 | 1] → Bottom

Heap as { Complete Binary Tree }



∵ fill the left element first. then right.

(insertion in Max Heap)

```
    parent = i/2 } index.
if (arr[parent] > arr[index]) return;
else swap(arr[parent], arr[index]);
    index = parent;
```

deletion:

```
swap[arr[root], arr[last]]
delete last
if(left < size &&
        arr[i] < arr[left]
swap(arr[i], arr[left])
  i = left ; }
else if( right < size && arr[i] < arr[right])
{
    swap( arr[i], arr[right]); i = right ; }
```

Page No.

Date ___/___/_____

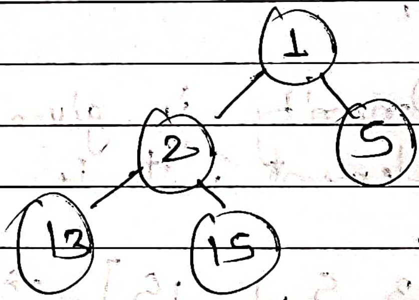
Min Heap – Priority is given to the smaller element. It is sorted in ascending order of elements.

ex → [2, 13, 5, 1, 15].

Array implementation of Heap.

Top. → | 1 | 2 | 5 | 13 | 15 | → Bottom.

Heap as {Complete Binary Tree}



Operations in Heap.

1. Top → It returns the topmost element of the heap. if empty() then throw an error T.C → O(1)

2. Push → It inserts an element into the heap. The size increases by one. T.C → O(logn) 'n' number of elements inside the heap already.

3. Pop → Topmost element poped from the heap. Size decreases by one. If empty() throw an error. T.C. → (Log n)

4. Size → Return size of the heap. T.C. → O(1) if empty return 0.

etc.

Date ___/___/______

As I already said Heaps are complete Binary Tree. Refer to Binary Tree section for more about CBT.

* Conversion of Array into Heap.
Let us consider the given elements to be in a form of an array.

1. The first element is taken as the root.
2. Let the element at index 'i' in the array be kept on the current root.
3. Then, the left child of that root will be the element at index '2*i+1'
4. Then, the right child of that root will be the element at index, '2*i+2'.

5. In this way only, the whole complete binary tree is createad.

6. Index of array considered to be start from '0'

if there is '1' indexed array, then.

left → 2*i , right → 2*i+1

* Declaration of Heap → { C++ STL }.

1. For Max Heap → priority_queue<int> max_pq;

2. For Min Heap → priority_queue<int, vector<int>, greater<int>>

priority_queue<int, vector<int>, greater<int>> min_pq;

Page No.

Date ___/___/______

Application → 'k' based question.

1. Whenever we are asked to find the kth smallest or largest element in the array. we can use heaps

1. One method is to sort the array & return arr[k-1] element T.C → $O(n\log n)$.

2. By using heap. we can insert the element in the heap one by one. As soon as the size of heap become greater than 'k' we pop the element T.C. $O(n\log k)$.

3. For kth smallest element we use max heap.
4. For kth Largest " " min heap.

2. Priority Based question → Some priority Linked in a question.

1. Returning min/max element at each query after performing some operations after each query.

2. Reserving the minimum element number query wise.

3. Huffman Coding → Heaps are usually used in Huffman coding while encoding & decoding the encrypted string message.